

```
Compiled from "Hello.java" public class Hello minor version: 0 major
version: 52 flags: ACC_PUBLIC ACC_SUPER Constant pool: #1 = Methodref
#6.#15 // java/lang/Object.<init>:()V #2 = Methodref #16.#17 //
java/lang/System.out:Ljava/io/PrintStream; #3 = String #18 // Hello world!
#4 = Methodref #19.#20 // java/io/PrintStream.println:(Ljava/lang/String;)V
#5 = Class #21 // Hello #6 = Class #22 // java/lang/Object #7 = Utf8
<init> #8 = Utf8 ()V #9 = Utf8 Code #10 = Utf8 LineNumberTable #11 =
Utf8 main #12 = Utf8 ([Ljava/lang/String;)V #13 = Utf8 SourceFile #14 =
Utf8 Hello.java #15 = NameAndType #7:#8 // <init>:()V #16 = Class #23
// java/lang/System.out:Ljava/io/PrintStream; #17 = Utf8 out
#18 = Utf8 Hello world! #19 = Class #26 // java/io/PrintStream #20 =
NameAndType #27:#28 // println:(Ljava/lang/String;)V #21 = Utf8 Hello #22
= Utf8 java/lang/Object #23 = Utf8 java/lang/System #24 = Utf8 out #25
= Utf8 Ljava/io/PrintStream; #26 = Utf8 java/io/PrintStream #27 = Utf8
println #28 = Utf8 (Ljava/lang/String;)V { public Hello(); descriptor: ()V
flags: ACC_PUBLIC Code: stack=2, locals=1, args_size=1 0: aload_0 1:
invokespecial #1 // Method java/lang/Object.<init>:()V 4: return
LineNumberTable: line 1: 0 public static void main(java.lang.String[]);
descriptor: ([Ljava/lang/String;)V flags: ACC_PUBLIC, ACC_STATIC Code:
stack=2, locals=1, args_size=1 0: getstatic #2 // Field
java/lang/System.out:Ljava/io/PrintStream; 3: ldc #3 // String Hello world!
5: invokevirtual #4 // Method java/io/PrintStream.println:
(Ljava/lang/String;)V 8: return
SourceFile: "Hello.java"
LineNumberTable: line 3: 0 line 4: 8 }
```



УНИВЕРСИТЕТ ИТМО

# Программирование. 2 семестр

## Введение



- ✓ Дженирики и Коллекции
- ✓ Шаблоны проектирования
- ✓ Ввод-вывод
- ✓ Функциональное программирование
- ✓ Сетевое взаимодействие
- ✓ Многопоточность
- ✓ Работа с базами данных
- ✓ Графический интерфейс и локализация

✓ <https://se.ifmo.ru>

- задания к лабораторным
- видео и тексты лекций
- методички

✓ <https://docs.oracle.com/javase/8/docs/api/>

✓ <https://docs.oracle.com/en/java/javase/17/docs/api/>



УНИВЕРСИТЕТ ИТМО

# Программирование. 2 семестр

## Обобщения

```
Compiled from "Hello.java" public class Hello minor version: 0 major
version: 52 flags: ACC_PUBLIC ACC_SUPER Constant pool: #1 = Methodref
#6.#15 // java/lang/Object; #2 = Methodref #16.#17 //
java/lang/System.out: Ljava/io/PrintStream; #3 = String #18 // Hello world!
#4 = Methodref #19.#20 // java/io/PrintStream.println:(Ljava/lang/String;)V
#5 = Class #21 // Hello #6 = Class #22 // java/lang/Object #7 = Utf8
<init> #8 = Utf8 ()V #9 = Utf8 Code #10 = Utf8 LineNumberTable #11 =
Utf8 main #12 = Utf8 ([Ljava/lang/String;)V #13 = Utf8 SourceFile #14 =
Utf8 Hello.java #15 = NameAndType #7:#8 // <init>:()V #16 = Class #23
// java/lang/System.out: Ljava/io/PrintStream; #17 = Utf8 out
#18 = Utf8 Hello world! #19 = Class #26 // java/io/PrintStream #20 =
NameAndType #27:#28 // println:(Ljava/lang/String;)V #21 = Utf8 Hello #22
= Utf8 java/lang/Object #23 = Utf8 java/lang/System #24 = Utf8 out #25
= Utf8 Ljava/io/PrintStream; #26 = Utf8 java/io/PrintStream #27 = Utf8
println #28 = Utf8 (Ljava/lang/String;)V { public Hello(); descriptor: ()V
flags: ACC_PUBLIC Code: 0: aload_0 1:
invokespecial #1 // Method java/lang/Object.<init>:()V 4: return
LineNumberTable: line 1: 0 public static void main(java.lang.String[]);
descriptor: ([Ljava/lang/String;)V flags: ACC_PUBLIC, ACC_STATIC Code:
stack=2, locals=1, args_size=1 0: getstatic #2 // Field
java/lang/System.out: Ljava/io/PrintStream; 3: ldc #3 // String Hello world!
5: invokevirtual #4 // Method java/io/PrintStream.println:
(Ljava/lang/String;)V 8: return
SourceFile: "Hello.java"
ITMO
UNIVERSITY
```

- ✓ Тип данных
  - диапазон значений + операции
- ✓ Типобезопасность
  - Проверка совместимости типа
- ✓ Статическая и динамическая типизация
  - момент связывания переменной с типом
- ✓ Сильная и слабая типизация
  - явное и неявное приведение

## ☑ Статическая типизация

- тип переменной при объявлении
- компилятор контролирует типы
- оптимизация на уровне машинных кодов

## ☑ Динамическая типизация

- тип есть у значения, у переменной - при присваивании
- код проще и более гибкий
- больше ошибок

## ☑ Сильная

- минимум неявного приведения
- запрет операций над несовместимыми типами

## ☑ Слабая

- неявное приведение
- операции с разными типами
- JavaScript
  - ◆ WAT - Gary Bernhardt, 2012
  - ◆ <https://www.destroyallsoftware.com/talks/wat>

## ☑ Примитивные

- значение
- не наследник Object
- эффективные
- операции
- массивы
- не обобщается

## ☑ Ссылочные

- ссылка
- наследник Object
- больше и медленнее
- методы
- массивы и коллекции
- обобщается



✓ int

- byte
- short
- long

✓ знаковые в дополнительном коде

$$0111(7) + 1 = 1000 (-8)$$

$$1111(-1) + 1 = 0000 (0)$$

✓ double

- float

✓ IEEE 754      знак-порядок-мантисса

0, -0, Infinity, -Infinity, NaN

✓ boolean

✓ true, false

✓ char

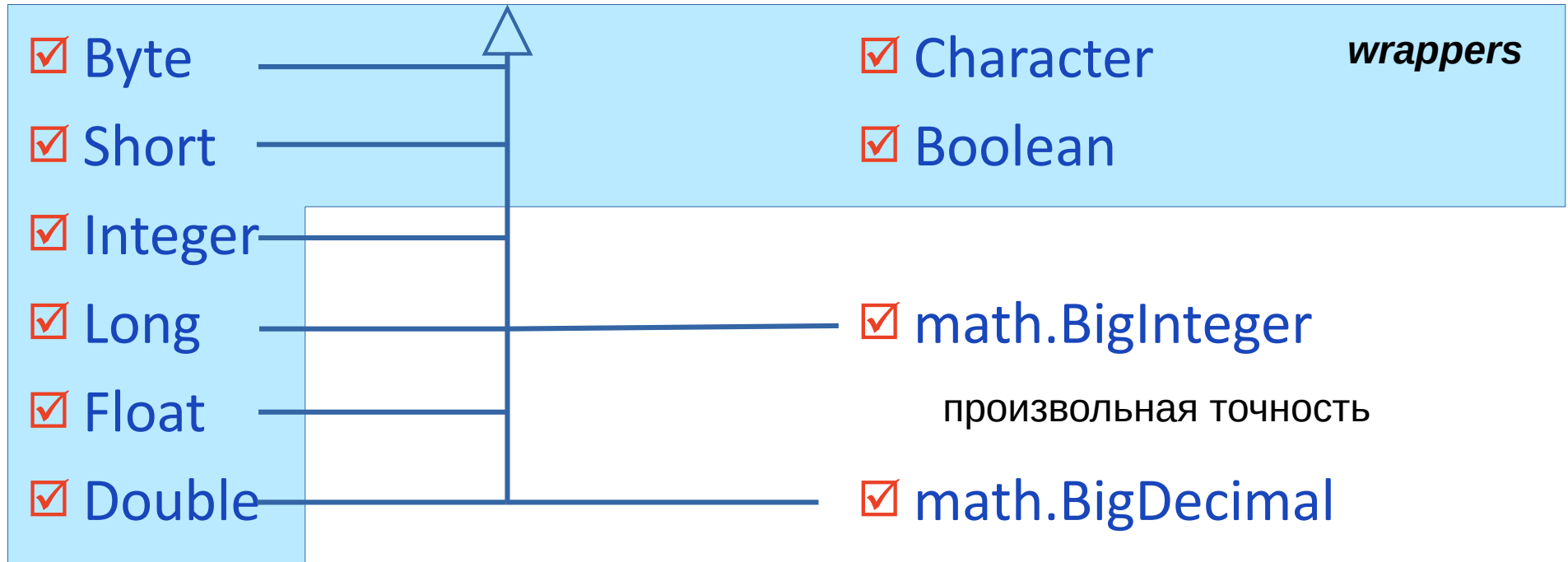
✓ Unicode, 16 бит, беззнаковые целые



- ✓ ссылка на адрес в памяти (32 бита)
- ✓ массив
  - адрес начала + размер элемента \* индекс
- ✓ строка (String) - массив символов
- ✓ объект - произвольный ADT (Abstract Data Type)
  - class
  - interface

# Классы-оболочки (Wrappers)

☑ Number



# Integer и int

---

```
int prim = 2023;
```

```
Integer wrap = new Integer(2023);
```

```
int prim = 2023;
```

```
Integer wrap = new Integer(2023);
```

```
Integer wrap = Integer.valueOf(2023);
```

```
Integer w2 = Integer.valueOf(prim);
```

```
int p2 = w2.intValue();
```

```
int prim = 2023;
```

```
Integer wrap = Integer.valueOf(2023);
```

```
wrap = 2023;
```

```
Integer w2 = Integer.valueOf(prim);
```

```
w2 = prim;
```

```
int p2 = w2.intValue();
```

```
p2 = w2;
```

```
int prim = 2023;
```

Autoboxing / автоупаковка

```
Integer wrap = Integer.valueOf(2023);    wrap = 2023;
```

```
Integer w2 = Integer.valueOf(prim);      w2 = prim;
```

Autounboxing / автораспаковка

```
int p2 = w2.intValue();                  p2 = w2;
```

```
int prim = 2023;
```

```
w2 += 888;
```

```
w2 = Integer.valueOf(w2.intValue()+888);
```

Autoboxing / автоупаковка

```
wrap = 2023;
```

```
w2 = prim;
```

Autounboxing / автораспаковка

```
p2 = w2;
```



# Integer и int

```
Integer w1 = 999;
```

```
Integer w2 = 999;
```

```
w1.equals(w2)
```

```
w1 == w2
```

# Integer и int

```
Integer w1 = 999;
```

```
Integer w2 = 999;
```

```
w1.equals(w2)    // true
```

```
w1 == w2        // false
```

```
Integer w1 = 999;
```

```
Integer w2 = 999;
```

```
w1.equals(w2)    // true
```

```
w1 == w2        // false
```

```
Integer w1 = 100;
```

```
Integer w2 = 100;
```

```
w1.equals(w2)    // true
```

```
w1 == w2        // ?
```

```
Integer w1 = 999;
```

```
Integer w2 = 999;
```

```
w1.equals(w2) // true
```

```
w1 == w2 // false
```

```
Integer w1 = 100;
```

```
Integer w2 = 100;
```

```
w1.equals(w2) // true
```

```
w1 == w2 // true
```

- ✓ Стандарт Unicode (было 16 бит, стало - 21)
- ✓ char (16 бит) U+0000 ... U+FFFF - UTF-16 code unit
  - U+D800 .. U+DBFF - high surrogate
  - U+DC00 .. U+DFFF - low surrogate
- ✓ int (21 бит) U+0000 ... U+10FFFF - code point
- ✓ Методы:
  - Character valueOf(char), char charValue()
  - chars[] toChars(int), int toCodePoint(char high, char low)

- ☑ Обобщенное программирование
  - абстрактное описание данных и алгоритмов, применяемое без изменения к разным типам данных
- ☑ Параметрический полиморфизм
  - тип данных является параметром
- ☑ Дополнительный контроль на этапе компиляции
  - более безопасный код

# Пример без обобщений - Box

```
public class Box {  
    private Object obj;  
  
    public Box(Object o) {  
        obj = o;  
    }  
  
    public Object get() {  
        return obj;  
    }  
}
```



```
public class Box {  
    private Object obj;  
  
    public Box(Object o) {  
        obj = o;  
    }  
  
    public Object get() {  
        return obj;  
    }  
}  
  
Box box = new Box("Hello!");  
  
int len = box.get().length();
```



# Тест 1 - get() возвращает Object

```
public class Box {  
    private Object obj;  
  
    public Box(Object o) {  
        obj = o;  
    }  
  
    public Object get() {  
        return obj;  
    }  
}
```

```
Box box = new Box("Hello!");  
  
int len = box.get().length();
```

```
error: cannot find symbol  
symbol:   method length()  
location: class Object
```

```
public class Box {  
  
    private Object obj;  
  
    public Box(Object o) {  
        obj = o;  
    }  
  
    public Object get() {  
        return obj;  
    }  
}
```

```
Box box = new Box("Hello!");  
  
String o = box.get();  
int len = o.length();
```

```
public class Box {  
  
    private Object obj;  
  
    public Box(Object o) {  
        obj = o;  
    }  
  
    public Object get() {  
        return obj;  
    }  
}
```

```
Box box = new Box("Hello!");
```

```
String o = box.get();  
int len = o.length();
```

```
error: incompatible types: Object  
cannot be converted to String  
        String o = box.get();  
                           ^
```

```
public class Box {  
    private Object obj;  
    public Box(Object o) {  
        obj = o;  
    }  
    public Object get() {  
        return obj;  
    }  
}
```

```
Box box = new Box("Hello!");  
String o = (String) box.get();  
int len = o.length();
```

# Тест 3 - успешная компиляция

```
public class Box {  
    private Object obj;  
    public Box(Object o) {  
        obj = o;  
    }  
    public Object get() {  
        return obj;  
    }  
}  
  
Box box = new Box(256);  
  
String o = (String) box.get();  
int len = o.length();  
  
// компиляция без ошибок!
```



# Тест 3 - исключение приведения типа

```
public class Box {  
  
    private Object obj;  
  
    public Box(Object o) {  
        obj = o;  
    }  
  
    public Object get() {  
        return obj;  
    }  
}
```

```
Box box = new Box(256);
```

```
String o = (String) box.get();  
int len = o.length();
```

```
Exception in thread "main"  
java.lang.ClassCastException:  
class java.lang.Integer cannot be cast  
to class java.lang.String
```

```
public class Box {  
  
    private Object obj;  
  
    public Box(Object o) {  
        obj = o;  
    }  
  
    public Object get() {  
        return obj;  
    }  
}
```

```
public class Box<T> {  
  
    private T obj;  
  
    public Box(T o) {  
        obj = o;  
    }  
  
    public T get() {  
        return obj;  
    }  
}
```

T — параметр типа

# Тест 4 - успешная компиляция

```
public class Box<T> {  
    private T obj;  
  
    public Box(T o) {  
        obj = o;  
    }  
  
    public T get() {  
        return obj;  
    }  
}
```

```
Box<String> box =  
    new Box<>("Hello");  
  
String o = (String) box.get();  
int len = o.length();
```

Обобщенный тип  
(generic)

Параметризованный тип  
(parametrized)



```
public class Box<T> {  
    private T obj;  
  
    public Box(T o) {  
        obj = o;  
    }  
  
    public T get() {  
        return obj;  
    }  
}
```

```
Box<String> box =  
    new Box<>(256);  
  
String o = box.get();  
int len = o.length();
```

```
public class Box<T> {  
    private T obj;  
  
    public Box(T o) {  
        obj = o;  
    }  
  
    public T get() {  
        return obj;  
    }  
}
```

```
Box<String> box =  
    new Box<>(256);
```

```
String o = box.get();  
int len = o.length();
```

```
error: incompatible types: Integer  
cannot be converted to String  
    box.put(s);  
           ^
```

- ✓ E — элемент коллекции (в Java API)
- ✓ K — ключ
- ✓ V — значение
- ✓ N — число
- ✓ T — первый параметр типа
- ✓ S, U, V — второй, третий, четвертый параметр типа

## ☑ C++ - templates

- **разный код** для каждого параметра
- во время исполнения информация о шаблонах **недоступна**

## ☑ C# - generics

- **разный код** для каждого **примитивного** параметра
- во время исполнения информация об обобщениях **доступна**

## ☑ Java - generics

- при компиляции создается **код без обобщений** (type erasure)
- во время исполнения информация об обобщениях **недоступна**

✓ Совместимость  
со старыми версиями

✓ Недостаточная  
типобезопасность

- Контроль только при компиляции
- При выполнении возможны нарушения

✓ Нельзя делать:

- Обобщенный примитив
  - ◆ `Box<int>`
- Создавать объект параметра типа
  - ◆ `E e = new E()`
- Создавать массивы обобщений
  - ◆ `Box<Cat>[]`
- Перегружать методы с обобщениями
  - ◆ `void print(Box<Integer>)`
  - ◆ `void print(Box<String>)`

- ✓ Параметр типа перед типом возвращаемого значения

```
public static <T> Box<T> pack(T obj)
```

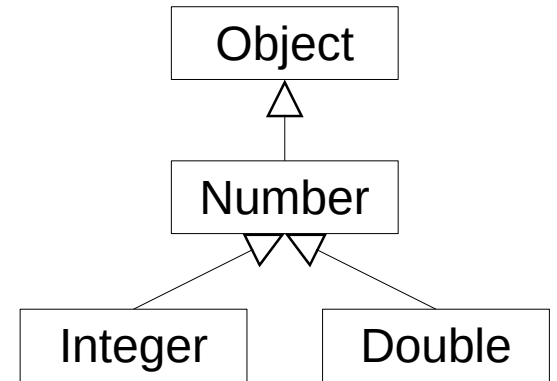
- ✓ При вызове метода нужен явный или неявный параметр

```
Box<Integer> b1 = Box.<Integer>pack(1);
```

```
Box<String> b2 = Box.pack("Hello");
```

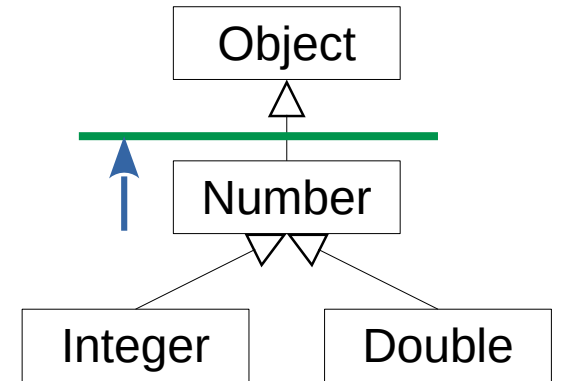
✓ `Box<T>`, `T` - любой тип (потомок `Object`)

```
public class NumBox<T> {  
    private Object obj  
  
    public int incAndGet() {  
        obj += 1;  
        return obj;  
    }  
}
```

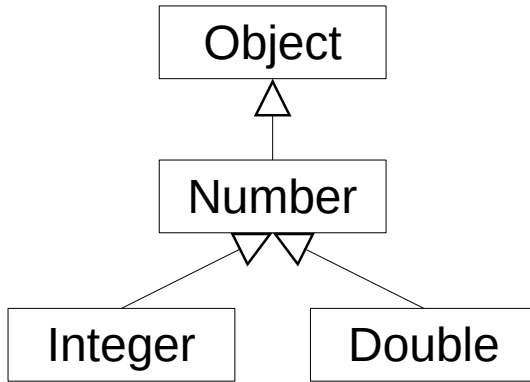


## ☑ Инкремент только для чисел

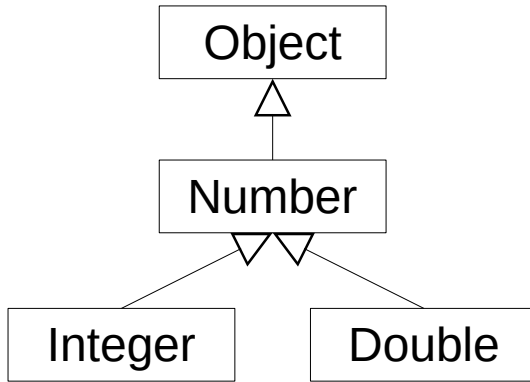
```
public class NumBox<T extends Number> {  
    private T obj  
  
    public int incAndGet() {  
        return obj.intValue() + 1;  
    }  
}
```



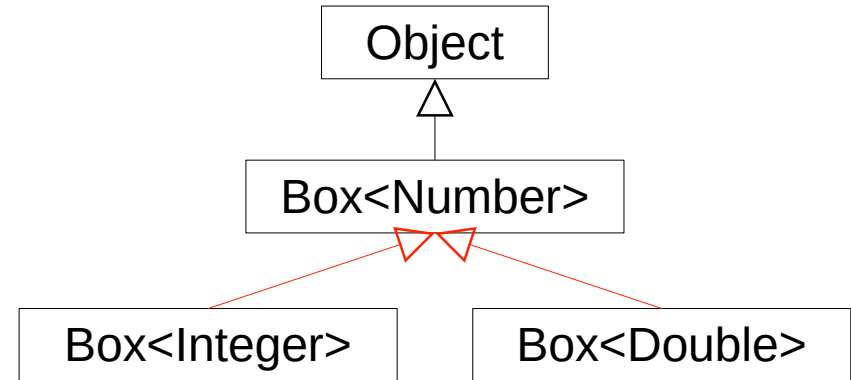
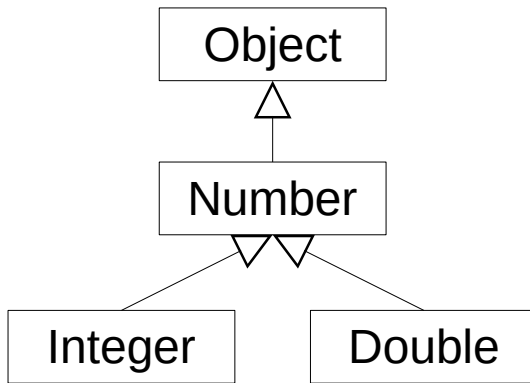




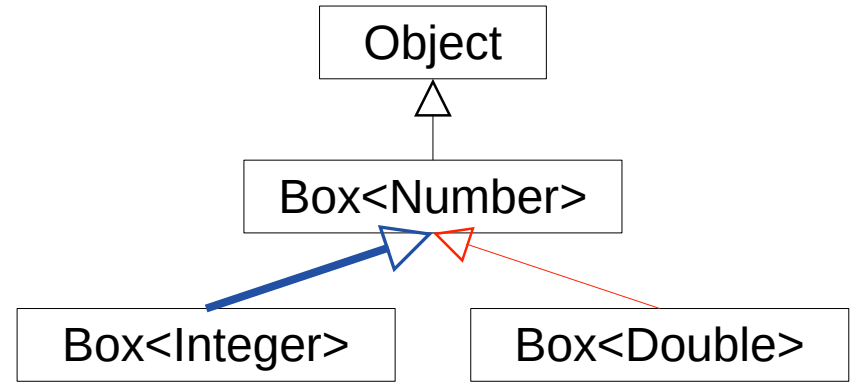
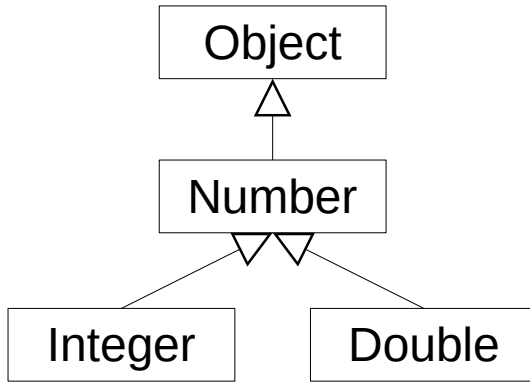
```
Number num;
Integer i = 511;
Double d = 3.1416;
num = i;
num = d;
```



```
Box<Number> bnum;
Integer i = 511;
Double d = 3.1416;
bnum = new Box<>(i);
bnum = new Box<>(d);
```



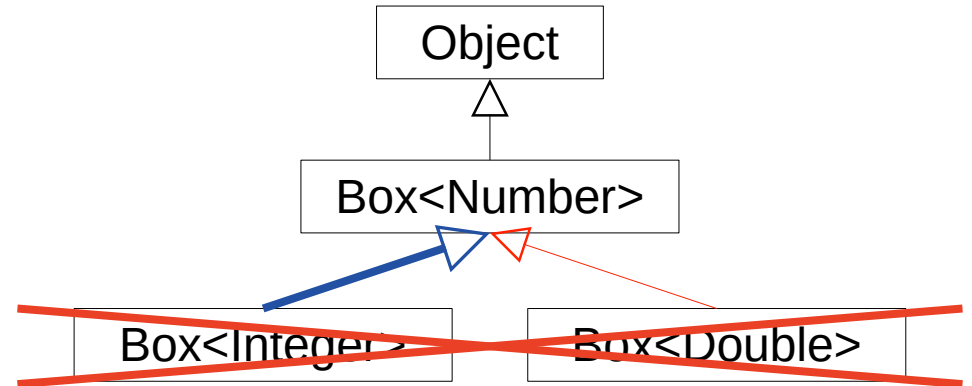
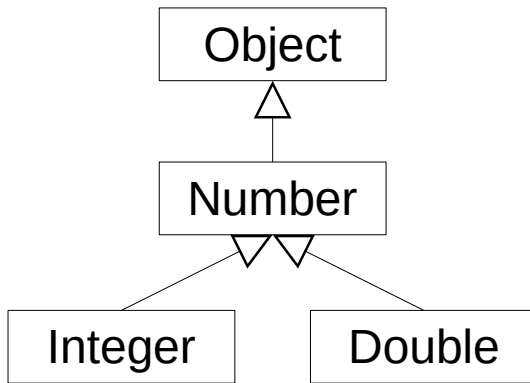
```
Box<Integer> bint;  
Integer i = 511;  
Double d = 3.1416;  
bint = new Box<>(i);  
bint = new Box<>(d); // cannot infer type arguments for Box<>  
bint = new Box<Double>(d); // Box<Double> cannot be converted to Box<Integer>  
bint = new Box<Integer>(d); // Double cannot be converted to Integer
```



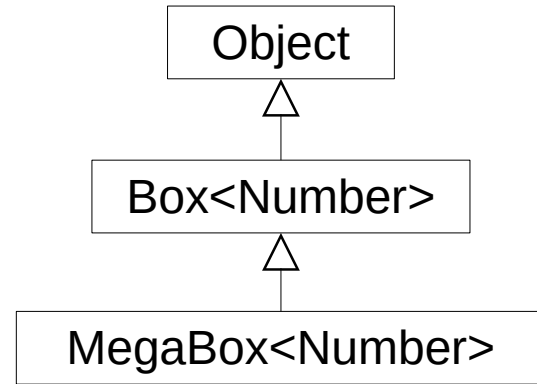
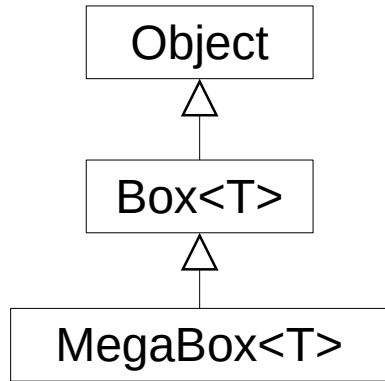
```
Box<Integer> bint;  
Integer i = 511;  
Double d = 3.1416;  
Box<Number> bnum = (Box<Number>) bint;
```

```
bnum = new Box<>(i);  
bnum = new Box<>(d);
```

# Тест 6 - несовместимые типы



```
Box<Integer> bint;  
Integer i = 511;  
Double d = 3.1416;  
Box<Number> bnum = (Box<Number>) bint;  
// Box<Integer> cannot be converted to Box<Number>  
bnum = new Box<>(i);  
bnum = new Box<>(d);
```



```
Box<Number> bnum;  
MegaBox<Number> mega; // MegaBox extends Box
```

```
mega = new MegaBox<>(3, 1415926535897932384626433832795);  
bnum = mega;
```

# Обобщенный класс - параметр метода

```
public class NBox<N extends Number> extends Box<N> {  
  
    public NBox(N o) {  
        super(o);  
    }  
  
    public double avg(NBox<N> box) {  
        double v = box.get().doubleValue();  
        return (box.get().doubleValue() + v) / 2.0;  
    }  
}
```

```
NBox<Double> box1 = new NBox<>(1.0);  
NBox<Double> box2 = new NBox<>(5.0);  
System.out.println(box1.avg(box2));
```

# Обобщенный класс - параметр метода

```
public class NBox<N extends Number> extends Box<N> {  
  
    public NBox(N o) {  
        super(o);  
    }  
  
    public double avg(NBox<N> box) {  
        double v = box.get().doubleValue();  
        return (box.get().doubleValue() + v) / 2.0;  
    }  
}
```

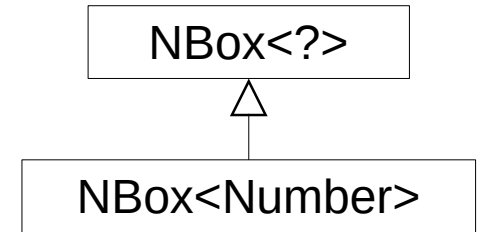
```
NBox<Double> box1 = new NBox<>(1.0);  
NBox<Integer> box2 = new NBox<>(5);  
System.out.println(box1.avg(box2));
```



# Подстановки (wildcards)

```
public class NBox<N extends Number> extends Box<N> {  
  
    public double avg(NBox<? extends Number> box) {  
        double v = box.get().doubleValue();  
        return (box.get().doubleValue() + v) / 2.0;  
    }  
}
```

```
NBox<Double> box1 = new NBox<>(1.0);  
NBox<Integer> box2 = new NBox<>(5);  
System.out.println(box1.avg(box2));
```

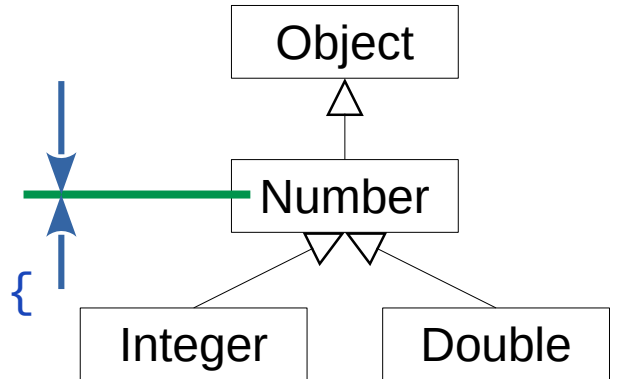


## ✓ class List<T>

- public double sum(List<? extends Number> list)
- public void print(List<?> list)
- public <T> void fill(List<? super T> list, T value)

## ✓ Class Enum<E extends Enum<E>>

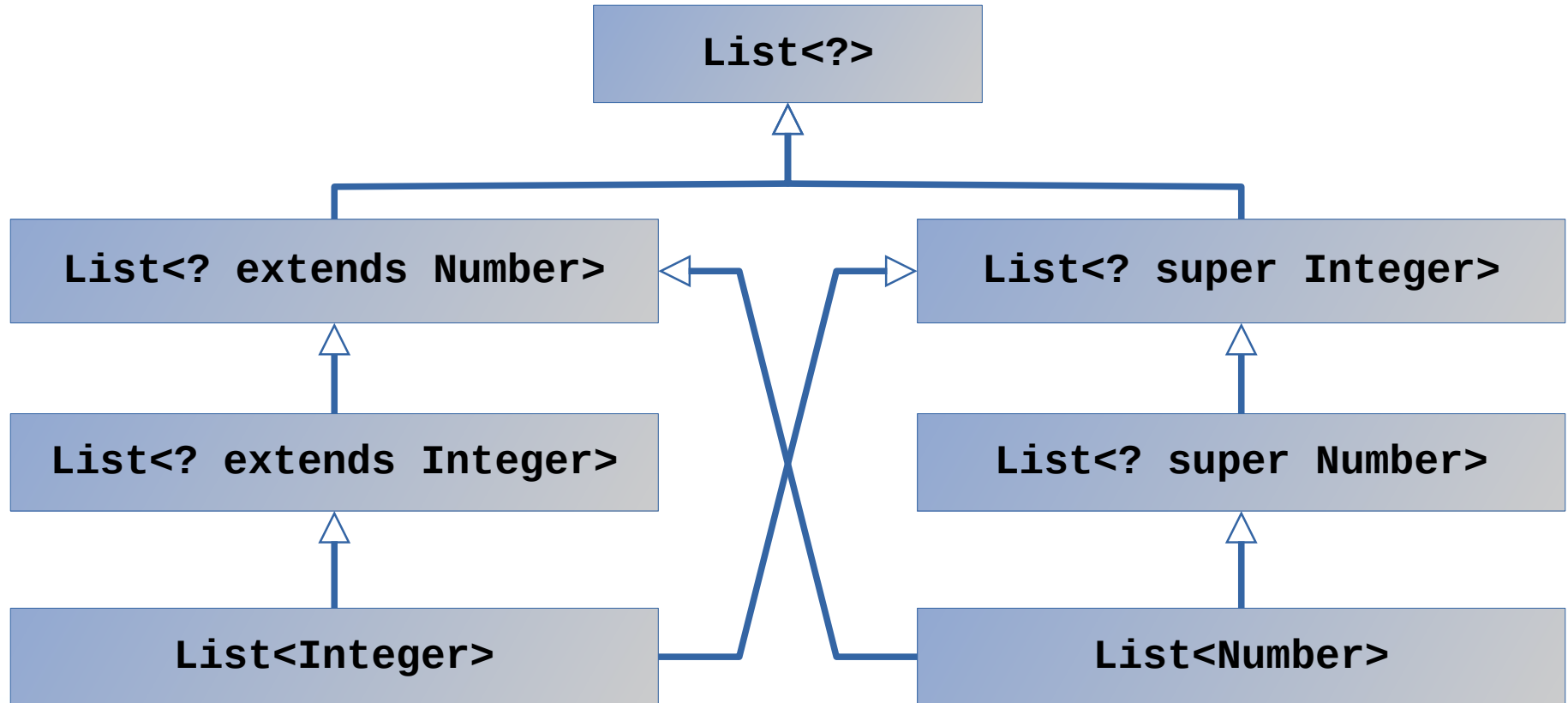
```
public class Box<T> {  
    private T obj;  
  
    public void copyTo(Box<? super T> to) {  
        to.obj = obj;  
    }  
    public void copyFrom(Box<? extends T> from) {  
        obj = from.obj;  
    }  
}  
  
Box<Number> nbox = new Box<>();  
Box<Integer> ibox = new Box<>(42);  
ibox.copyTo(nbox);    // T = Integer, ? = Number  
nbox.copyFrom(ibox); // T = Number, ? = Integer
```



- ✓ Параметры методов - производители (producers) и потребители (consumers)
- ✓ Правило PECS - Producer Extends, Consumer Super
- ✓ Производители ограничиваются сверху `<? extends X>`
- ✓ Потребители ограничивается снизу `<? super X>`
- ✓ Производители, ограниченные классом Object - использовать неограниченную подстановку `<?>`
- ✓ Параметр одновременно потребитель и производитель — подстановки не используются (тип задается явно)

- ☑ Наследование массивов - ковариантно
  - Integer[] потомок Number[]
- ☑ Наследование обобщений - инвариантно
  - Box<Integer> не потомок и не предок Box<Number>
- ☑ Ограничение сверху - ковариантно
  - Box<? extends Number>
- ☑ Ограничение снизу - контрвариантно
  - Box<? super Number>

# Схема наследования обобщений





УНИВЕРСИТЕТ ИТМО

# Программирование. 2 семестр

## Коллекции

```
Compiled from "Hello.java" public class Hello {
  minor version: 0 major version: 52
  flags: ACC_PUBLIC ACC_SUPER
  Constant pool: #1 = Methodref #6.#15 // java/lang/Object.<init>:()V #2 = Methodref #16.#17 //
  java/lang/System.out:Ljava/io/PrintStream; #3 = String #18 // Hello world!
  #4 = Methodref #19.#20 // java/io/PrintStream.println:(Ljava/lang/String;)V
  #5 = Class #21 // Hello #6 = Class #22 // java/lang/Object #7 = Utf8
  <init> #8 = Utf8 ()V #9 = Utf8 Code #10 = Utf8 LineNumberTable #11 =
  Utf8 main #12 = Utf8 ([Ljava/lang/String;)V #13 = Utf8 SourceFile #14 =
  Utf8 Hello.java #15 = NameAndType #7:#8 // <init>:()V #16 = Class #23
  // java/lang/System.out:Ljava/io/PrintStream; #17 = Utf8 Hello world!
  #18 = Utf8 Hello world! #19 = Class #26 // java/io/PrintStream #20 =
  NameAndType #27:#28 // println:(Ljava/lang/String;)V #21 = Utf8 Hello #22
  = Utf8 java/lang/Object #23 = Utf8 java/lang/System #24 = Utf8 out #25
  = Utf8 Ljava/io/PrintStream; #26 = Utf8 java/io/PrintStream #27 = Utf8
  println #28 = Utf8 (Ljava/lang/String;)V { public Hello(); descriptor: ()V
  flags: ACC_PUBLIC Code: s args_size=1 0: aload_0 1:
  invokespecial #1 // Method java/lang/Object.<init>:()V 4: return
  LineNumberTable: line 1: 0 public static void main(java.lang.String[]);
  descriptor: ([Ljava/lang/String;)V flags: ACC_PUBLIC, ACC_STATIC Code:
  stack=2, locals=1, args_size=1 0: getstatic #2 // Field
  java/lang/System.out:Ljava/io/PrintStream; 3: ldc #3 // String Hello world!
  5: invokevirtual #4 // Method java/io/PrintStream.println:
  (Ljava/lang/String;)V 8: return
  SourceFile: "Hello.java"
  LineNumberTable: line 3: 0 line 4: 8 }
```

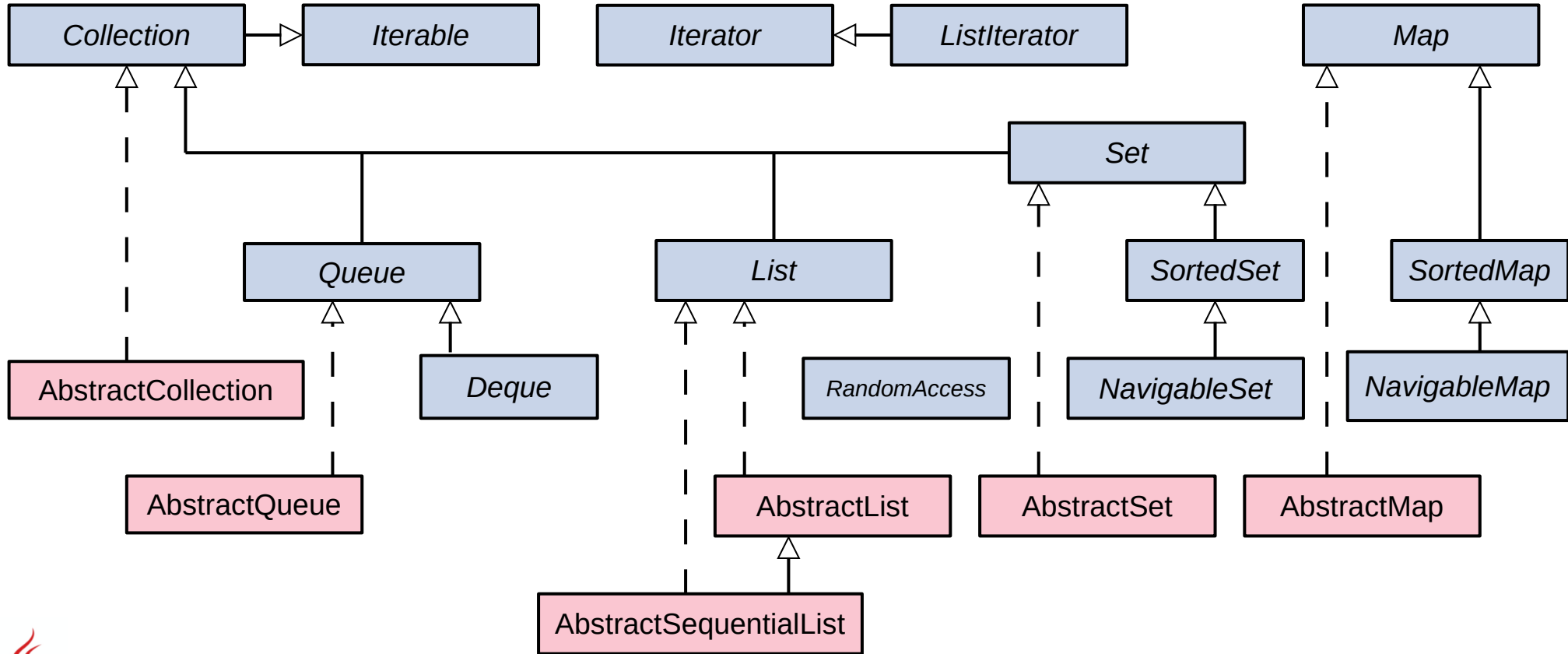


- ✓ Интерфейсы коллекций
- ✓ Абстрактные коллекции
- ✓ Реализации общего назначения
- ✓ Специальные реализации
- ✓ Потокобезопасные реализации
- ✓ Блокирующие очереди
- ✓ Вспомогательные классы



- ☑ Базовые интерфейсы определяют контракт
  - Что и как можно делать с элементами
  - Queue, Deque, List, Set, SortedSet, Map, SortedMap, ...
- ☑ Базовые реализации определяют характеристики
  - Скорость, занимаемая память, упорядоченность
  - Array, Linked List, Hash Table, Tree, Heap, ...
  - Big O-нотация

# Основные интерфейсы и абстракции



- ✓ Collection - коллекция одинаковых элементов
- ✓ Map - ассоциативный массив, пары "ключ-значение"
- ✓ SortedMap - сортировка по ключам
- ✓ List - последовательность индексированных элементов
- ✓ Set - множество уникальных элементов
- ✓ SortedSet - сортировка по значениям
- ✓ Queue - очередь, элементы входят и выходят
- ✓ Deque - двусторонняя очередь

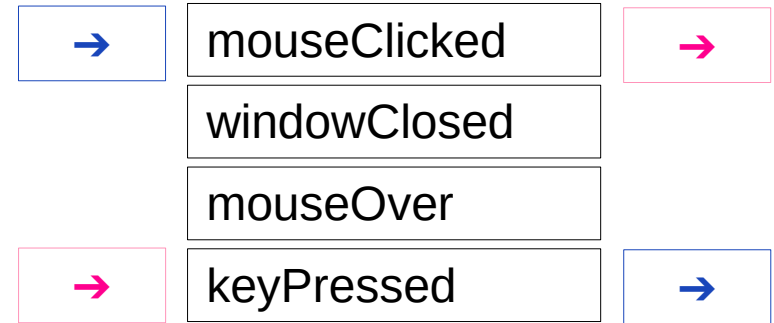
## ☑ List

0	яблоко
1	апельсин
2	банан
3	КИВИ

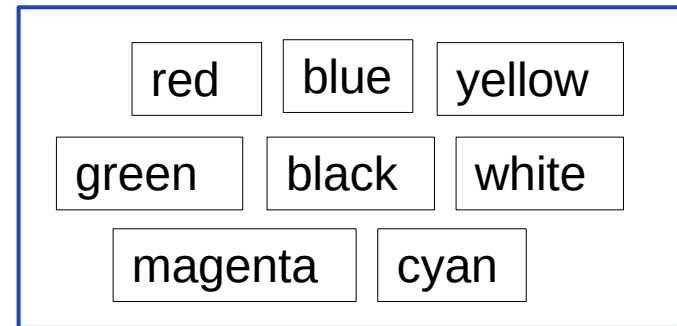
## ☑ Map

и	37153142
т	31620970
м	16203060
о	55414481

## ☑ Queue/Deque/Stack



## ☑ Set



# Интерфейс Iterable<T>

- ✓ Итерируемый объект - обход всех элементов
- ✓ Цикл `for (x : Iterable)`
- ✓ `Iterator<T> iterator()`

☑ Обход элементов коллекции

☑ Методы:

- boolean hasNext()
- E next()
- void remove()

☑ Пример

```
Iterator<Short> it = c.iterator();  
while (it.hasNext()) {  
    System.out.println(it.next());  
}
```

## ✓ boolean add (E e)

- результат - элемент есть в коллекции
- true - коллекция изменилась

## ✓ boolean remove (Object o)

- результат - элемента нет в коллекции
- true - коллекция изменилась

## ✓ boolean contains (Object o)

- true - элемент есть, false - элемента нет

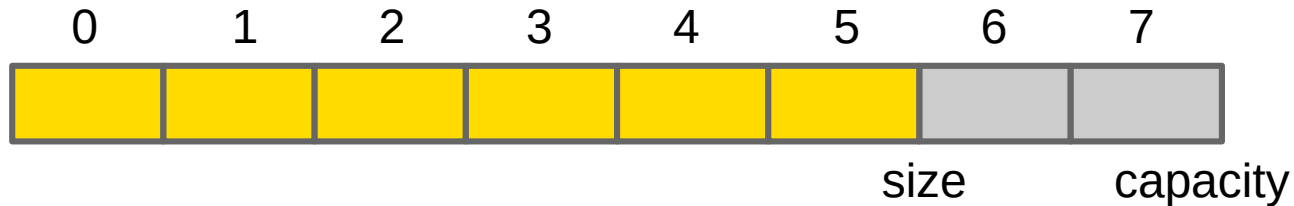
- ✓ `addAll(Collection c)`
- ✓ `removeAll(Collection c)`
- ✓ `retainAll(Collection c)`
- ✓ `containsAll(Collection c)`
  
- ✓ `clear()`
- ✓ `isEmpty()`
- ✓ `size()`



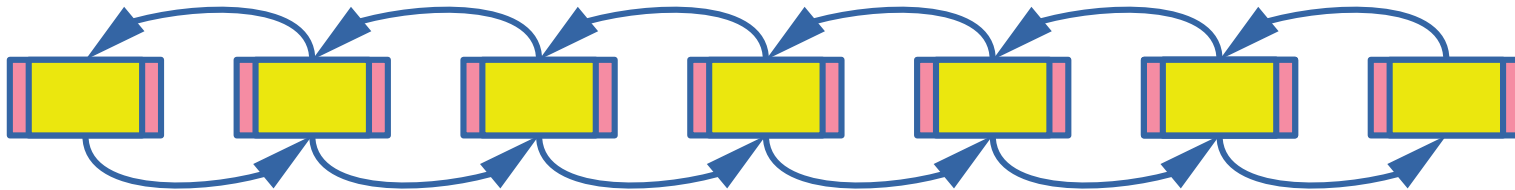


- ✓ Индекс от 0 до N
- ✓ E get(int index)
- ✓ E set(int index, E e)
- ✓ void add(int index, E e)
- ✓ E remove(int index)
  - List<Integer> list
  - list.remove(1) ???

- ✓ Реализация списка (list) на основе массива (array)
- ✓ ArrayList extends AbstractList implements List, RandomAccess
- ✓ Быстрый произвольный доступ по индексу



- ✓ Реализация на основе двусвязного списка
- ✓ LinkedList extends AbstractSequentialList implements List
- ✓ Последовательный доступ к элементам
- ✓ Быстрое добавление и удаление



## ☑ Vector

- потокобезопасный ArrayList

## ☑ Stack

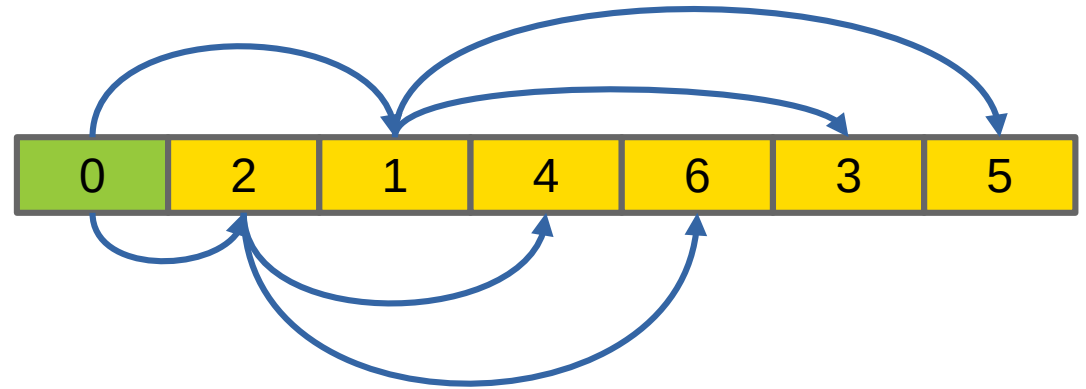
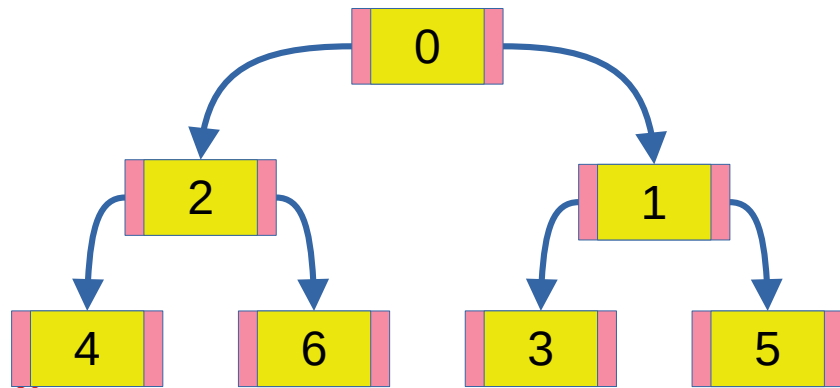
- устаревшая реализация на базе вектора

## ☑ CopyOnWriteArrayList

- новая копия при каждом изменении

- ✓ Как правило - FIFO (first in, first out)
- ✓ Добавляем в конец, забираем из начала
- ✓ Методы, бросающие исключения
  - `boolean add()` , `E remove()`, `E element()`
- ✓ Методы, возвращающие особые значения
  - `boolean offer()`, `E poll()`, `E peek()`

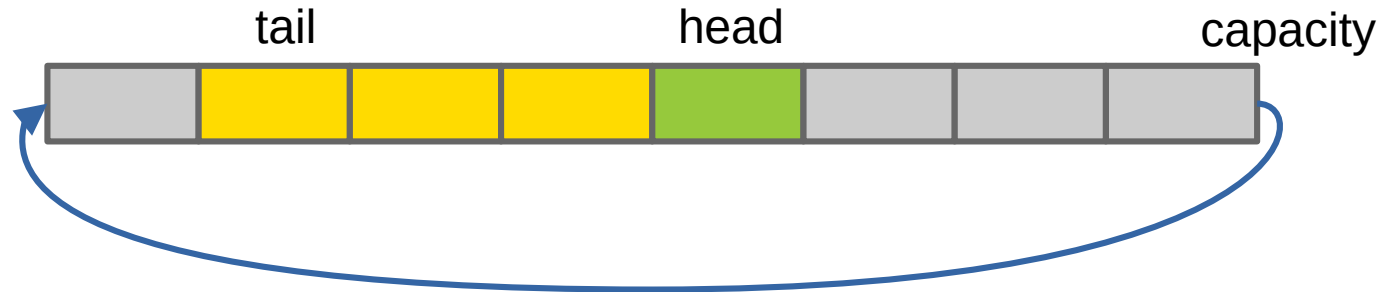
- ✓ Реализация на основе кучи (heap)
- ✓ Heap - дерево, корень - наименьший элемент
- ✓ FIFO + очередь тех, кто без очереди
- ✓ Нужна реализация сравнения элементов



- ✓ Двусторонняя очередь
- ✓ Может быть стеком
- ✓ Методы
  - addFirst, removeFirst, getFirst
  - addLast, removeLast, getLast
  - offerFirst, pollFirst, peekFirst
  - offerLast, pollLast, peekLast

# Дек на основе массива - ArrayDeque

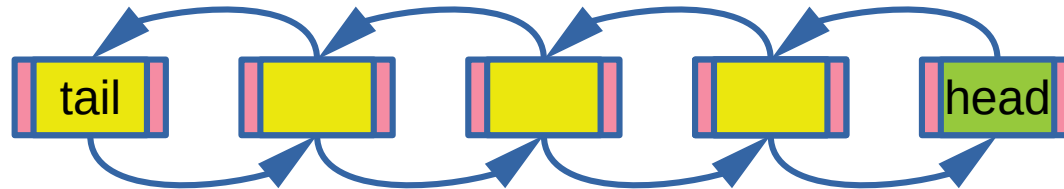
- ✓ Реализация на основе массива
- ✓ Быстрое добавление и удаление -  $O(1)$





# И снова двусвязный список

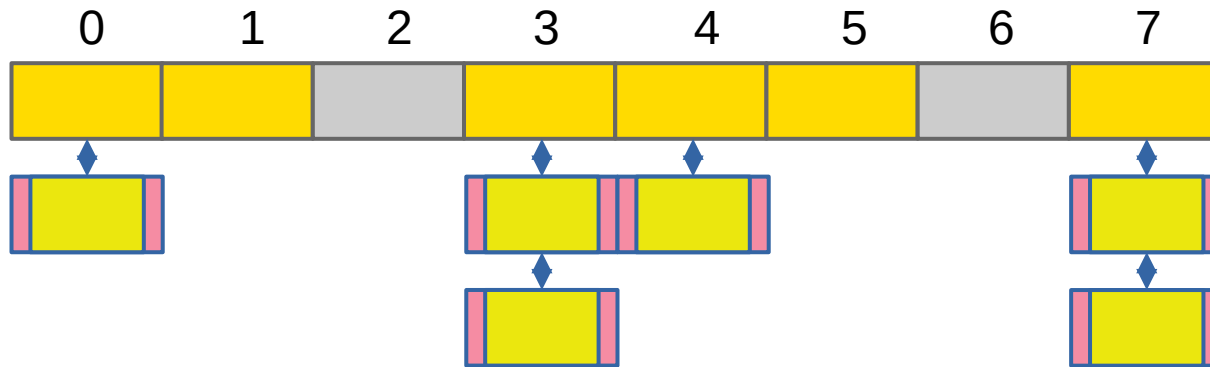
- ✓ LinkedList implements Queue, Deque, List
- ✓ Быстрые добавление, удаление -  $O(1)$ ,



- ✓ Два параметра типа
- ✓ Ключи уникальные, значения - любые
- ✓ Методы:
  - `V put(K key, V value)`
  - `V get(Object K)`
  - `V remove(Object k)`
- ✓ Представления:
  - `keySet()`, `values()`, `entrySet()`

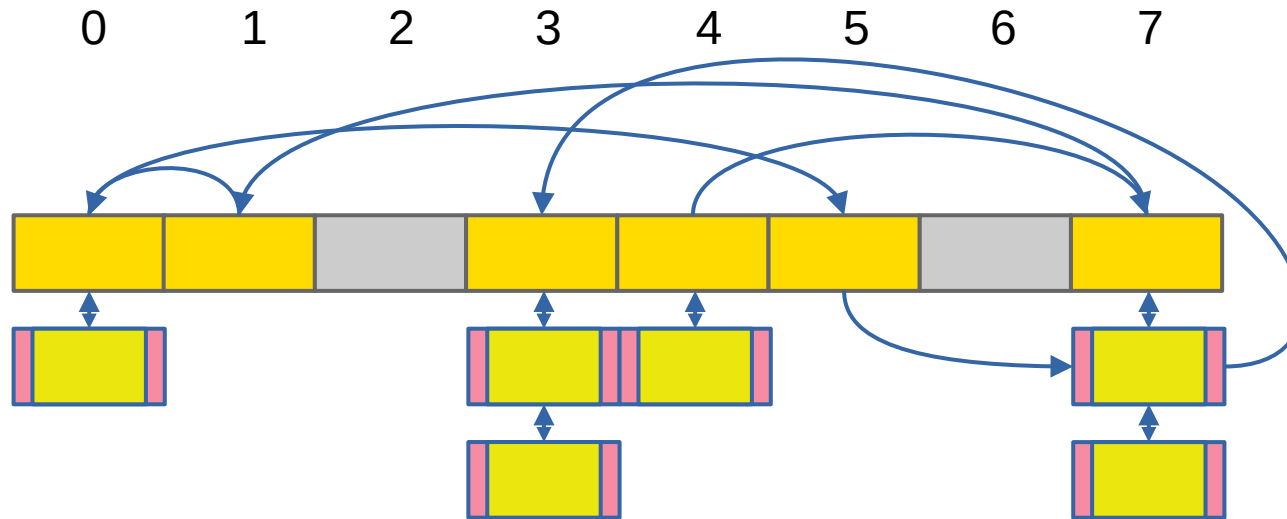
- ✓ Ассоциативный массив на основе хеш-таблицы
- ✓  $\text{bucket} = \text{key.hashCode()} \% N$
- ✓ Нужен метод `hashCode()` у элементов
- ✓ Обход элементов - в полном беспорядке

N buckets



- ✓ HashMap + связный список
- ✓ Получение элементов в порядке добавления

N buckets



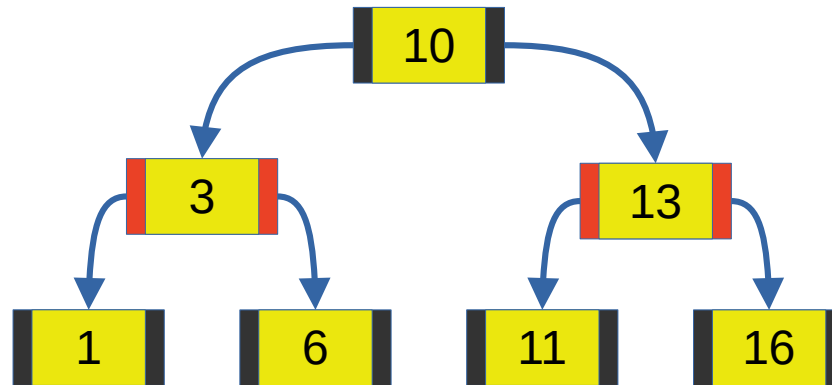
## ✓ SortedMap<K, V>

- K firstKey(), K lastKey()
- SortedMap subMap(K from, K to)
- SortedMap headMap(K k), SortedMap tailMap(K k)

## ✓ NavigableMap<K,V>

- lowerKey, floorKey, ceilingKey, higherKey (<, <=, >= >)
- lowerEntry, floorEntry, ceilingEntry, higherEntry
- subMap, headMap, tailMap - (K k, boolean inclusive)

- ✓ Реализация на основе Red-Black Tree
- ✓ Самобалансирующееся бинарное дерево
- ✓ Элементы отсортированы
- ✓ Нужна реализация сортировки



## ☑ Hashtable

- синхронизированная реализация

## ☑ WeakHashMap

- ключи - слабые ссылки

## ☑ EnumMap

- ключи - enum, реализация на основе массива

## ☑ IdentityHashMap

- ключи одинаковые только в случае идентичности

- ☑ Методы как у Collection
- ☑ Массовые операции - объединение, пересечение, разность множеств
- ☑ Реализации - как у Map
  - HashSet implements Set,
  - LinkedHashSet extends HashSet,
  - TreeSet implements SortedSet, NavigableSet
  - ключи являются значениями



## ☑ CopyOnWriteArraySet

- новая копия при изменении

## ☑ EnumSet

- реализация на основе битовой карты

- ✓ ArrayList vs LinkedList
- ✓ HashSet vs LinkedHashSet vs TreeSet
- ✓ HashMap vs LinkedHashMap vs TreeMap
- ✓ PriorityQueue vs LinkedList vs ArrayDeque

## ☑ коллекции-обертки:

- synchronized
- unmodifiable
- checked

## ☑ алгоритмы для List

- sort(), shuffle(), reverse(), fill(), swap(), binarySearch()

## ☑ фабрики коллекций

- emptySet, emptyList, emptyMap,
- singleton, singletonList, singletonMap

## ☑ и еще много полезного



- ☑ Методы для работы с массивами (обычными)
  - сортировка, поиск, копирование, заполнение
  - и еще много полезного



УНИВЕРСИТЕТ ИТМО

# Программирование. 2 семестр

## Сравнение элементов

```
Compiled from "Hello.java" public class Hello minor version: 0 major
version: 52 flags: ACC_PUBLIC ACC_SUPER Constant pool: #1 = Methodref
#6.#15 // java/lang/Object: <init>:()V #2 = Methodref #16.#17 //
java/lang/System.out: Ljava/io/PrintStream; #3 = String #18 // Hello world!
#4 = Methodref #19.#20 // java/io/PrintStream.println:(Ljava/lang/String;)V
#5 = Class #21 // Hello #6 = Class #22 // java/lang/Object #7 = Utf8
<init> #8 = Utf8 ()V #9 = Utf8 Code #10 = Utf8 LineNumberTable #11 =
Utf8 main #12 = Utf8 ([Ljava/lang/String;)V #13 = Utf8 SourceFile #14 =
Utf8 Hello.java #15 = NameAndType #7:#8 // <init>:()V #16 = Class #23
// java/lang/System.out: Ljava/io/PrintStream; #17 = Utf8 out
#18 = Utf8 Hello world! #19 = Class #26 // java/io/PrintStream #20 =
NameAndType #27:#28 // println:(Ljava/lang/String;)V #21 = Utf8 Hello #22
= Utf8 java/lang/Object #23 = Utf8 java/lang/System #24 = Utf8 out #25
= Utf8 Ljava/io/PrintStream; #26 = Utf8 java/io/PrintStream #27 = Utf8
println #28 = Utf8 (Ljava/lang/String;)V { public Hello(); descriptor: ()V
flags: ACC_PUBLIC ACC_STATIC Code: 0: aload_0 1:
invokespecial #1 // java/lang/object: <init>:()V 4: return
LineNumberTable: line 1: 0 public static void main(java.lang.String[]);
descriptor: ([Ljava/lang/String;)V flags: ACC_PUBLIC, ACC_STATIC Code:
stack=2, locals=1, args_size=1 0: getstatic #2 // Field
java/lang/System.out: Ljava/io/PrintStream; 3: ldc #3 // String Hello world!
5: invokevirtual #4 // Method java/io/PrintStream.println:
(Ljava/lang/String;)V 8: return
SourceFile: "Hello.java"
ITMO
UNIVERSITY
```

- ☑ Интерфейсы `Comparable<T>` и `Comparator<T>`
  - основной метод возвращает `int` (`< 0`, `== 0`, `> 0`)
- ☑ `java.lang.Comparable<T>` ( `x.compareTo(y)` )
  - естественный порядок сортировки
  - реализуется при создании класса
  - реализован в большинстве библиотечных классов
- ☑ `java.util.Comparator<T>` ( `c.compare(x, y)` )
  - любой необходимый порядок сортировки
  - объект создается по мере необходимости

- 1) Выбираем два элемента
  - 2) Как-то сравниваем - Comparable или Comparator
  - 3) При необходимости перемещаем элементы
- 
- ☑ Готовые эффективные алгоритмы сортировки
  - ☑ Все, что нужно - предоставить метод сравнения

# Сортировка (естественная)

---

```
public class A {  
    List<String> list;  
    public void do() {  
        Collections.sort(list);  
        for (String s : list) { System.out.println(s); }  
    }  
}
```

class String implements Comparable



# Сортировка (отдельный компаратор)

```
public class A {
    List<String> list;
    public void do() {
        Collections.sort(list, new C());
        for (String s : list) { System.out.println(s); }
    }
}

class C implements Comparator<String> {
    public int compare(String s1, String s2) {
        return s1.length() - s2.length();
    }
}
```



# Сортировка (статический вложенный класс)

```
public class A {  
    List<String> list;  
    public void do() {  
        Collections.sort(list, new C());  
        for (String s : list) { System.out.println(s); }  
    }  
    static class C implements Comparator<String> {  
        public int compare(String s1, String s2) {  
            return s1.length() - s2.length();  
        }  
    }  
}
```



# Сортировка (внутренний класс)

```
public class A {  
    List<String> list;  
    public void do() {  
        Collections.sort(list, new A().new C());  
        for (String s : list) { System.out.println(s); }  
    }  
    class C implements Comparator<String> {  
        public int compare(String s1, String s2) {  
            return s1.length() - s2.length();  
        }  
    }  
}
```



# Сортировка (локальный класс)

```
public class A {  
    List<String> list;  
    public void do() {  
        class C implements Comparator<String> {  
            public int compare(String s1, String s2) {  
                return s1.length() - s2.length();  
            }  
        }  
        Collections.sort(list, new C());  
        for (String s : list) { System.out.println(s); }  
    }  
}
```



# Сортировка (анонимный класс)

```
public class A {  
    List<String> list;  
    public void do() {  
        Collections.sort(list, new Comparator<String>() {  
            public int compare(String s1, String s2) {  
                return s1.length() - s2.length();  
            }  
        });  
        for (String s : list) { System.out.println(s); }  
    }  
}
```



# Сортировка (лямбда-выражение)

```
public class A {  
    List<String> list;  
    public void do() {  
        Collections.sort(list,  
            (s1, s2) -> s1.length() - s2.length());  
        for (String s : list) { System.out.println(s); }  
    }  
}
```

# Программирование. 2 семестр

## Лабораторные

ITMO More than a  
UNIVERSITY

- ☑ Внимательно прочитать задание и подумать
  - Посмотреть любой вариант лаб 6, 7 и 8
- ☑ Выделить классы (не забывать SOLID)
  - Подумать над общей структурой
- ☑ Нарисовать диаграмму
  - Объяснить уточке, как это должно работать
- ☑ Написать маленький работающий код
  - Дописывать до победы



- ☑ Command (getName(), getDescr(), execute() )
  - HelpCommand
  - ShowCommand
  - AddCommand
  
- ☑ Map<String, Command>
  - map.put(command.getName(), command)
  - ...
  - map.get(str[0]).execute();